

# A Communication-Efficient Parallel Method for Group-Lasso

Binghong Chen, Jun Zhu  
Tsinghua University, Beijing

cbh13@mails.tsinghua.edu.cn, dcszj@mail.tsinghua.edu.cn

## Abstract

Group-Lasso (gLasso) identifies important explanatory factors in predicting the response variable by considering the grouping structure over input variables. However, most existing algorithms for gLasso are not scalable to deal with large-scale datasets, which are becoming a norm in many applications. In this paper, we present a divide-and-conquer based parallel algorithm (DC-gLasso) to scale up gLasso in the tasks of regression with grouping structures. DC-gLasso only needs two iterations to collect and aggregate the local estimates on subsets of the data, and is provably correct to recover the true model under certain conditions. We further extend it to deal with overlappings between groups. Empirical results on a wide range of synthetic and real-world datasets show that DC-gLasso can significantly improve the time efficiency without sacrificing regression accuracy.

## 1 Introduction

In many regression problems, we are interested in identifying important explanatory factors in predicting the response variable. Lasso [Tibshirani, 1996] represents a type of widely applied methods with sound theoretical guarantee. To consider the settings where each explanatory factor may be represented by a group of derived input variables, group-Lasso (gLasso) has been developed [2006], which yields group-wise sparse estimates. gLasso has been applied in various applications, including multifactor analysis-of-variance (ANOVA) problem [Yuan and Lin, 2006], learning pairwise interactions between regression factors [Lim and Hastie, 2013], solving EEG source problems arising from visual activation studies [Lim, 2013], estimating breeding values using molecular markers spread over the whole genome [Ogutu and Piepho, 2014], visual saliency detection [Souly and Shah, 2015] and functional MRI data analysis [Shimizu *et al.*, 2015].

Many algorithms have been developed to solve the optimization problem of gLasso, including blockwise coordinate descent (BCD) [Yuan and Lin, 2006; Meier *et al.*, 2008], (fast) iterative shrinkage-thresholding algorithm (ISTA/FISTA) [Beck and Teboulle, 2009; Liu *et al.*, 2009; Villa *et al.*, 2014], hybrid BCD and ISTA [Qin *et al.*, 2013],

alternating direction method (ADM) [Qin and Goldfarb, 2012] and groupwise-majorization-descent (GMD) [Yang and Zou, 2014]. Among these algorithms, GMD was reported to run faster than BCD and FISTA [Yang and Zou, 2014] on a single machine. However, with the fast-growing volume of datasets, the storage and computation of data in one single machine become difficult, and the need to design a scalable algorithm for regression with grouping structure is urgent. Previous work has been done to parallelize computation on the level of one iteration in an iterative procedure [Peng *et al.*, 2013]. Although this type of parallel computing can save computation time, the time reduced is often limited due to the frequent communication between machines.

In contrast, a divide-and-conquer (DC) approach performs parallel computing at the level of the whole optimization process, where the dataset is split into shards and each worker handles a subset of data locally to produce local estimates. Then, a master node collects the local estimates and combines them to obtain a final estimate according to some aggregation strategy. For linear models, *averaging* is the simplest and most popular strategy, which was proposed by McDonald *et al.* [2009] and carefully studied by Zinkevich *et al.* [2010] as well as Zhang, Duchi and Wainwright [2012]. Wang *et al.* [2014] adopt a similar idea and develop a two-step parallel inference algorithm for Lasso, where in the model selection phase, the variables are selected using majority voting from the local Lasso estimates; and in the coefficient estimation phase, the coefficients of the selected variables are estimated by averaging the local ordinary least square (OLS) estimates.

In this paper, we adopt the DC framework and present a parallel inference algorithm for gLasso (DC-gLasso) in the regression tasks with grouping structures among input variables. Similar to the algorithm proposed by Wang *et al.* [2014], our algorithm has two distributed computing steps — it aggregates the local sparse estimates by gLasso to select a subset of variables via majority voting at the first step and then averages the local coefficients estimated by OLS regression on the selected variables to get the final estimate. We show that our algorithm successfully scales up regression with grouped variables by theoretical analysis and experiment evidence. We further extend the method to deal with overlapping structures among feature groups [Jacob *et al.*, 2009].

The rest of the paper is organized as follows. We will formulate the problem and introduce group-Lasso in next Sec-

tion. In Section 3, we will present DC-gLasso in detail. Finally, we will present some theoretical analysis (Section 4) and empirical results (Section 5).

## 2 Problem formulation and group-Lasso

Let  $\mathbf{X}$  denote a given  $n \times p$  design matrix, where  $n$  is the number of observations and  $p$  is the number of features. We consider the linear regression model, where the response variables  $Y \in \mathbb{R}^n$  are modeled as:

$$Y = \mathbf{X}\beta + \epsilon, \quad (1)$$

$\beta$  is a  $p$ -dimensional coefficient vector, and  $\epsilon$  is the Gaussian white noise vector with mean 0 and variance  $\sigma^2$ .

An optimal weight vector  $\beta$  can be learned by minimizing a squared error with some regularization, e.g.,  $L_2$ -norm in ridge regression. In many scenarios, we expect to have a sparse model (i.e., most of the elements of  $\beta$  are zero), which is useful to identify a set of important explanatory factors for predicting the response variables and meanwhile protect the model from over-fitting. Various techniques have been developed for deriving a sparse estimate, with Lasso [Tibshirani, 1996] as one of the most extensively studied examples. Lasso performs an  $L_1$ -norm regularized linear regression problem and selects each individual elements.

In many applications, the variables are not independent. Instead, they may have some grouping structure. We would expect to select the variables at the group level. That is, a group of correlated variables is selected or discarded at the same time. Formally, let  $q$  denote the number of groups, and  $d_i$  denote the size of group  $i$ . So  $d_i$  must satisfy the condition that  $\sum_{i=1}^q d_i = p$ . Accordingly, the design matrix can be partitioned into  $q$  sub-matrices  $\mathbf{X}_i \in \mathbb{R}^{n \times d_i}$ , and the linear regression model can be restated as:

$$Y = \sum_{i=1}^q \mathbf{X}_i \beta_i + \epsilon, \quad (2)$$

where  $\beta_i$  is the corresponding  $d_i$ -dimensional coefficient vector for the variables in group  $i$ . The groupwise sparsity means that the groupwise vector  $\beta_i$  is zero or not. To obtain the above sparsity, group-Lasso (gLasso) [Yuan and Lin, 2006] solves the  $L_{2,1}$ -norm regularized linear regression problem:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \|Y - \mathbf{X}\beta\|_2^2 + \lambda \sum_{i=1}^q \|\beta_i\|_2, \quad (3)$$

where  $\lambda$  is a positive regularization parameter to control the sparsity level of the estimates.

The gLasso problem (3) is convex. Several solvers have been developed, including the classical Block Coordinate Descent (BCD) method [Yuan and Lin, 2006; Meier *et al.*, 2008], the gradient based ISTA/FISTA method and its extension [Beck and Teboulle, 2009; Liu *et al.*, 2009; Villa *et al.*, 2014], the extension and combination of BCD and ISTA [Qin *et al.*, 2013], the alternating direction method (ADM) [Qin and Goldfarb, 2012] and the groupwise-majorization-descent (GMD) algorithm [Yang and Zou, 2014]. Among these algorithms, GMD was reported to run much faster than BCD and FISTA [Yang and Zou, 2014].

However, all the above methods focus on solving the gLasso problem on a single machine. With the increase in the size of available data and the amount of exploitable computation resources, it is desirable to design a communication-efficient distributed algorithm to solve this optimization problem. In next section, we present a low-communication-cost parallel algorithm to solve the group-structured regression problem. As we shall see, our method is compatible with all the above single-machine algorithms to solve the sub-gLasso problems on each local machine.

## 3 A Divide-and-Conquer Algorithm

We now present a divide-and-conquer based parallel algorithm for group-Lasso. We build our work on the recent progress of MEdian Selection SubSet AGgregation Estimator (MESSAGE) [Wang *et al.*, 2014] for solving Lasso problems on large-scale data sets. MESSAGE is a divide-and-conquer algorithm that parallelizes Lasso selection, and it has excellent performance in variable selection, estimation, prediction, and computation time compared to alternative methods.

We adopt a similar divide-and-conquer approach in the regression tasks with grouping structure among variables. As outlined in Algorithm 1, our DC-gLasso algorithm consists of two stages — the *model selection* stage and the *coefficient estimation* stage. Each stage is performed in a divide-and-conquer framework, as explained below.

### 3.1 The model selection stage

Given a data set with  $n$  examples, we first split it randomly into  $m$  different subsets, and distribute them onto  $m$  machines. Now we have  $\frac{n}{m}$  samples on each machine. We will use  $(\mathbf{X}^k, Y^k)$  to denote the subset on machine  $k$ .

At the model selection stage, we perform gLasso on each machine for a fixed number of  $\lambda$  values. This can be done by any solver mentioned above. Then on the  $k^{th}$  machine, we select the optimal model  $\hat{M}_k$  through BIC criterion. Let  $\hat{\beta}^k$  denote the (sparse) estimate on machine  $k$ . We define

$$\hat{M}_k = \{i | \hat{\beta}_i^k \neq 0, i = 1 \cdots q\},$$

which denotes the set of groups that have non-zero weights in the local estimate on machine  $k$ . Once the master node collects all the local estimates  $\{\hat{M}_k : k = 1 \cdots m\}$ , it combines the selected models using majority voting to get the final sparse model  $\hat{M}$ , that is,

$$\hat{M} = \left\{ i \mid \sum_{k=1}^m 1_{i \in \hat{M}_k} \geq \frac{m}{2} \right\}, \quad (4)$$

where  $1_{i \in \hat{M}_k}$  is an indicator function that has value 1 if  $i \in \hat{M}_k$  and 0 otherwise. We can see that the final estimate  $\hat{M}$  selects feature group  $i$  if it is selected in no less than a half ( $m/2$ ) of the local models. Notice that, here, different from the original MESSAGE algorithm, the variables are selected in groups rather than in individuals.

### 3.2 The coefficient estimation stage

After we get the sparse pattern in the above step, we perform the coefficient estimation again in the divide-and-conquer setting, with the data split over  $m$  machines. In this stage, we

---

**Algorithm 1** A Parallel Inference Algorithm for gLasso

---

- 1: Split the data randomly into  $m$  subsets and distribute them on  $m$  machines.
- 2: **On each machine:** do local group-Lasso

$$\hat{\beta}^k = \underset{\beta}{\operatorname{argmin}} \|Y^k - \mathbf{X}^k \beta\|_2^2 + \lambda \sum_{i=1}^q \|\beta_i\|_2.$$

- 3: Vote for the best sparse pattern  $\hat{M}$  using the rule (4).
  - 4: **On each machine:** estimate the local linear regression weights  $\hat{\beta}_M^k$  by solving problem (5).
  - 5: Average to obtain the final estimate  $\hat{\beta}_{\hat{M}}$  via rule (6).
- 

first distribute the selected model  $\hat{M}$  to all the  $m$  machines. Then we perform OLS on the local data subset sitting on each machine. Notice that we can now throw away the variables which are not in model  $\hat{M}$ , so  $\beta_{\hat{M}}$  is estimated by:

$$\hat{\beta}_M^k = \underset{\beta_M^k}{\operatorname{argmin}} \|Y_M^k - \mathbf{X}_M^k \beta_M^k\|_2^2, \quad (5)$$

where  $\mathbf{X}_M^k$  denotes the part of the  $k^{th}$  design matrix (i.e.,  $\mathbf{X}^k$ ) that is selected by the sparse pattern  $\hat{M}$ , likewise for  $Y_M^k$ .

After the master node collects all the local weights  $\{\hat{\beta}_M^k : k = 1 \cdots m\}$ , the final weights are estimated by an average:

$$\hat{\beta}_{\hat{M}} = \frac{1}{m} \sum_{k=1}^m \hat{\beta}_M^k. \quad (6)$$

This step is the same as that in the MESSAGE algorithm.

### 3.3 Overlapping Group-Lasso

In real-world datasets, variables can often belong to more than one group, which leads to overlapping between groups. The overlapping group-Lasso [Jacob *et al.*, 2009] is an extension of gLasso to deal with these overlaps. It exploits the gLasso penalty with duplicated variables to obtain a sparse solution whose support (i.e., nonzero components of the recovered coefficients) is a union of groups.

Formally, let  $\nu_j \in \mathbb{R}^p$  denote a vector whose nonzero components are those positions corresponding to the features in group  $j$ , and let  $\mathcal{V}_j \subseteq \mathbb{R}^p$  be the subspace of such possible vectors. Then, by duplicating the variables in the original design matrix  $\mathbf{X}$ , we obtain a new matrix  $\mathbf{X}' = (\mathbf{X}_1, \dots, \mathbf{X}_q)$ , where  $\mathbf{X}_j$  is a sub-matrix of  $\mathbf{X}$  that corresponds to the  $j^{th}$  feature group. Note that there could be overlaps between  $\mathbf{X}_i$  and  $\mathbf{X}_j$  if groups  $i$  and  $j$  overlap. The coefficient vector  $\beta$  is given by  $\beta = \sum_{j=1}^q \nu_j$ , and the overlapping group-Lasso solves the optimization problem [Hastie *et al.*, 2015]:

$$\hat{\beta} = \underset{\beta = \sum_{j=1}^q \nu_j, \nu_j \in \mathcal{V}_j}{\operatorname{argmin}} \|Y - \mathbf{X}' (\sum_{j=1}^q \nu_j)\|_2^2 + \lambda \sum_{j=1}^q \|\nu_j\|_2.$$

This problem can be solved using a gLasso solver after duplicating the overlapped variables [Jacob *et al.*, 2009], and can be solved directly by accelerated gradient descent [Lei *et al.*,

---

**Algorithm 2** DC-gLasso with Overlap

---

- 1: Split the data randomly into  $m$  subsets and distribute them on  $m$  machines.
- 2: **On each machine:** do local overlapping group-Lasso

$$\hat{\beta}^k = \underset{\beta = \sum_{j=1}^q \nu_j, \nu_j \in \mathcal{V}_j}{\operatorname{argmin}} \|Y^k - \mathbf{X}'^k (\sum_{j=1}^q \nu_j)\|_2^2 + \lambda \sum_{j=1}^q \|\nu_j\|_2.$$

- 3: Vote for the best sparse pattern  $\hat{M}$  using the select-and-discard strategy (See text).
  - 4: **On each machine:** estimate the local linear regression weights  $\hat{\beta}_M^k$  by solving problem (5).
  - 5: Average to obtain the final estimate  $\hat{\beta}_{\hat{M}}$  via rule (6).
- 

2013], proximal gradient method [Argyriou *et al.*, 2011] and Alternating Direction Method of Multipliers (ADMM) [Boyd *et al.*, 2011].

We can extend DC-gLasso to perform parallel inference in the presence of group overlappings. The DC-gLasso with overlaps (DC-ogLasso) has the similar two-step procedure as DC-gLasso. Algorithm 2 outlines the procedure of DC-ogLasso. We can see that the key difference is in the model selection stage, where a distributed overlapping group-Lasso is carried out on each local machine using any solver as mentioned above, and instead of selecting features in groups, we adopt a *select-and-discard* strategy to select the sparse model. This strategy turns out to have a larger probability of selecting the correct model in practice compared with the select-in-groups strategy as in DC-gLasso.

The select-and-discard strategy works as follows. After the master node collects local estimates  $\hat{\beta}^k$ , a majority voting is applied to set each individual feature to get the sparse model  $\hat{M}$ . In order to ensure that the selected features constitute a union of groups, which is consistent with the property of overlapping group-Lasso, we apply “security check” to discard those features that are “alone” in model  $\hat{M}$ . By saying a feature is “alone”, we mean that there does not exist a group that contains the feature and whose features are all selected in model  $\hat{M}$ . Therefore, the discarding process ensures that we obtain a model whose support is a union of groups.

### 4 Model selection consistency

Wang *et al.* [2014] provide a nice model selection consistency bound for MESSAGE, based on previous work on theoretical analysis of Lasso [Zhao and Yu, 2006]. However, due to the complexity of various grouping structures of gLasso problems, it is very hard to derive a tight bound for gLasso, so is DC-gLasso. In this section, we provide a theoretical result on model selection consistency of our algorithm. Let  $\mathbf{M} = \{j | \beta_j \neq 0\}$  denote the sparsity pattern of the model parameter  $\beta$ . We make the same assumptions as [Bach, 2008]:

- (A)  $\mathbf{X}$  and  $Y$  have finite fourth order moments:  $\mathbb{E} \|\mathbf{X}\|^4 < \infty$  and  $\mathbb{E} \|Y\|^4 < \infty$ .
- (B) The joint matrix of second order moments  $\mathbb{E} \mathbf{X} \mathbf{X}^\top \in \mathbb{R}^{p \times p}$  is invertible.

(C) Strong condition for group-lasso consistency

$$\max_{i \in \mathbf{M}^c} \frac{1}{d_i} \left\| (\mathbf{X}_i^\top \mathbf{X}_\mathbf{M}) (\mathbf{X}_\mathbf{M}^\top \mathbf{X}_\mathbf{M})^{-1} \text{diag} \left( \frac{d_j}{\|\beta_j\|} \right) \beta_\mathbf{M} \right\| < 1,$$

where  $\mathbf{X}_\mathbf{M}$  is the sub-matrix of  $\mathbf{X}$  with the columns selected by  $\mathbf{M}$ , likewise for  $\beta_\mathbf{M}$  the sub-vector indexed by  $\mathbf{M}$ , and  $\text{diag}(\frac{d_j}{\|\beta_j\|})$  is the block-diagonal matrix with  $\frac{d_j}{\|\beta_j\|} I_{p_j}$  on the diagonal. Then, we have the following result:

**Theorem 1.** (Model Selection Consistency) *If each subset satisfies (A), (B) and (C), then there exist a sequence of  $\lambda$  such that the sparsity pattern of the estimates given by this algorithm  $M(\hat{\beta})$  converges in probability to  $\mathbf{M}$  when  $n/m$  goes to infinity.*

*Proof.* We use  $Pr_n$  to denote the probability of selecting the true model using group-Lasso with  $n$  samples. Then we have

$$\begin{aligned} \text{Prob}(\hat{M} = M) &\geq \text{Prob} \left( \sum_{i=1}^m 1_{\hat{M}_i = M} \geq \frac{m}{2} \right) \\ &\geq 1 - \frac{Pr_{n/m}(1 - Pr_{n/m})}{m(Pr_{n/m} - 1/2)^2}, \end{aligned}$$

where the first inequality holds due to the selection rule of  $\hat{M}$  in Eq. (4), and the second inequality holds due to Chebyshev's inequality. Under those model assumptions, we can apply Theorem 2 in [Bach, 2008], which states that there exist a sequence of  $\lambda$  such that  $Pr_n \rightarrow 1$  as  $n \rightarrow \infty$ . And  $\text{Prob}(\hat{M} = M) \rightarrow 1$  follows consequently.  $\square$

## 5 Experiments

We present experimental results on both synthetic and real data sets to demonstrate the effectiveness of our algorithm.

### 5.1 Synthetic datasets

We first present results on synthetic data sets, which were commonly used to evaluate sparse learning methods.

#### Performance

We design a simulation model and evaluate our algorithm based on its performance on datasets drawn from the model. The model is similar to the ones introduced in [Yuan and Lin, 2006] and [Yang and Zou, 2014]. The design matrix is generated as follows. The vectors  $Z_i, i = 1, 2, \dots, q$  are generated from a multivariate normal distribution with a correlation matrix such that the correlation between  $Z_i$  and  $Z_j$  is  $\rho$  for  $i \neq j$ . Let the  $s$ -sparse model's response vector  $Y$  be

$$Y = \sum_{s|i, 1 \leq i \leq q} \left( \frac{2}{3} Z_i - h_{i2} Z_i^2 + \frac{1}{3} h_{i3} Z_i^3 \right) t_i + k \epsilon_0,$$

where  $h_{i2}, h_{i3}$  are positive real numbers such that  $\|h_{i2} Z_i^2\|_2 = \|h_{i3} Z_i^3\|_2 = 1$ ;  $t_i = (-1)^{u_i} (3 + v_i)$ , with  $u_i$  randomly drawn from set  $\{0, 1\}$  and  $v_i$  drawn from  $\mathcal{N}(0, 1)$ ;  $\epsilon_0$  is a noise vector drawn from  $\mathcal{N}(0, 1)$  and a scaling parameter  $k$  ensures that the signal-to-noise ratio is 3.0. The summation condition  $s|i$  is true when  $i$  is a multiple of  $s$ , which

implies that only  $1/s$  variables are involved in the model, so we call it a  $s$ -sparse model.

When fitting it using a gLasso model, we treat the vectors  $\{Z_i, h_{i2} Z_i^2, h_{i3} Z_i^3\}$  as a group. Compared with Eq. (2), we have  $\mathbf{X}_i = (Z_i, h_{i2} Z_i^2, h_{i3} Z_i^3)$  and  $\beta_i = (\frac{2}{3} t_i, -t_i, \frac{1}{3} t_i)^\top$  when  $i$  is a multiple of  $s$ ,  $\mathbf{0}$  otherwise. So  $p = 3q$ , where  $p, q$  and  $n$  are the number of features, number of feature groups and sample size, respectively. The experiment task can be formulated as follows: given the response vector  $Y$  of length  $n$ , the design matrix  $\mathbf{X}$  of dimension  $n \times p$  and the grouping information (i.e., three consecutive features in one group), recover the sparsity pattern and value of model coefficient  $\beta$ .

We consider the following two combinations of  $q, s$  and  $\rho$ . In each setting, sample size  $n$  varies from 1,000 to 20,000.

**Scenario 1**  $(p, q, s, \rho) = (300, 100, 10, 0.5)$

**Scenario 2**  $(p, q, s, \rho) = (300, 100, 20, 0.5)$

In each scenario, we compare the performance of DC-gLasso with the counterpart algorithm on the full training set (denoted by gLasso). When performing DC-gLasso, we fix the subset size (i.e., the number of samples handled on one machine) at 1,000, which means that as the sample size grows from 1,000 to 20,000, the number of machines we use increases from 1 to 20. The full-set gLasso problem and the subset gLasso problems in our algorithm are solved by the "SGL" R-package, which implements the generalized gradient descent based algorithm [Simon *et al.*, 2013]. During each gLasso optimization, the best model is chosen using BIC from a solution path consisting twenty lambdas (the default setting of the "SGL" package).

We compare the performance in terms of computation time, mean square error between the recovered  $\hat{\beta}$  and the true  $\beta$ , and sparsity pattern of the recovered model (measured by the number of nonzero variables in  $\hat{\beta}$ ). The results are shown in Fig. 1 with "Fullset", "Parallel" and "Truth" denoting the result of full-set inference, the result of parallel inference and the ground truth, respectively. We can see that in both combinations of feature group number and sparsity, DC-gLasso obtains a huge reduction in computation time while doing a great job on recovering the sparsity pattern. With the inclusion of coefficient estimation stage, DC-gLasso is able to achieve a lower mean square error in estimating model coefficients. We can also see that DC-gLasso always selects fewer variables than the fullset inference method, but they achieve comparable results on recovering the model sparsity pattern.

To verify that our divide-and-conquer scheme can reduce computation time in grouped regression problems regardless of the gLasso solver, we repeat the experiment using another gLasso R-package "gglasso", which implements groupwise-majorization-descent algorithm [Yang and Zou, 2014], in the following settings:

**Scenario 3**  $(p, q, s, \rho) = (3000, 1000, 10, 0.5)$

**Scenario 4**  $(p, q, s, \rho) = (3000, 1000, 20, 0.5)$

Here the sample size  $n$  still varies from 1,000 to 20,000 with the subset size fixed at 1,000. During each gLasso optimization, the best model is chosen using BIC from a solution path consisting one hundred lambdas (the default setting of the "gglasso" package). The results are shown in Fig. 4. Still,

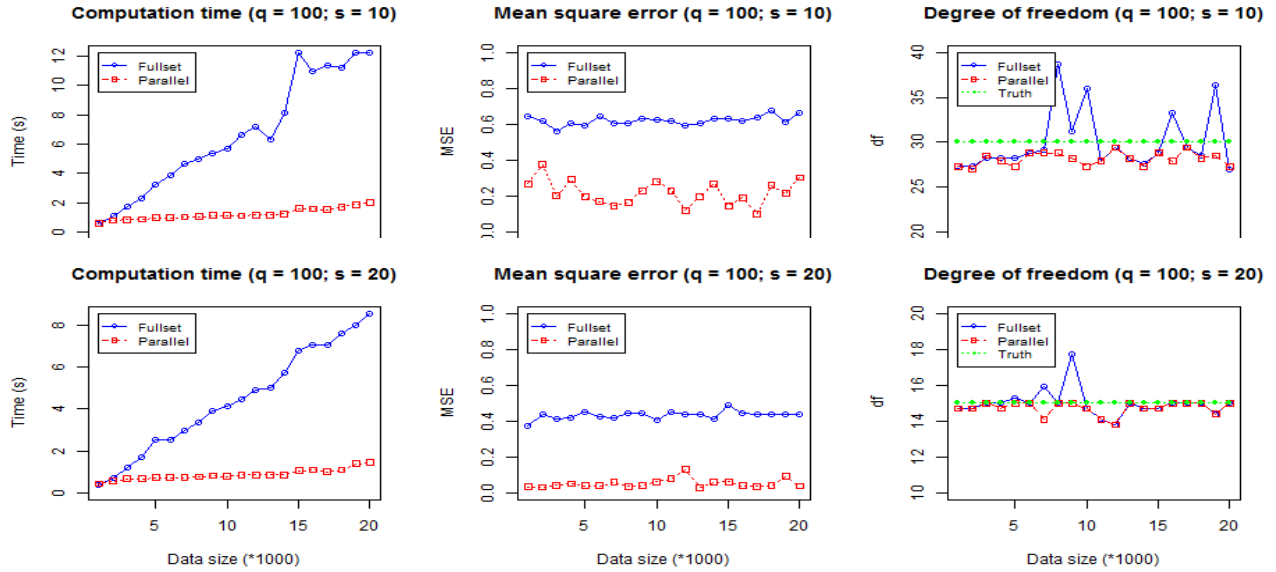


Figure 1: Results in **Scenarios 1 & 2** (group-Lasso part is implemented by R-package *SGL*).

our method offers nearly constant computing time when the sample size scales up while having lower mean square error and better recovering of sparsity pattern.

### Sample complexity measurement

In this experiment, we compare the degree of freedom of coefficients recovered by fullset gLasso and DC-gLasso to measure the number of samples needed to recover the true sparsity pattern for each method in the following two scenarios:

**Scenario 5**  $(m, p, q, s, \rho) = (10, 3000, 1000, 10, 0)$

**Scenario 6**  $(m, p, q, s, \rho) = (10, 3000, 1000, 20, 0)$

In both scenarios, 10 machines are used and the correlation between two feature vectors is set to zero to simplify the setting, with the subset size varying from 150 to 575. Fig. 2 shows that DC-gLasso needs 325 subset samples in scenario 5, where sparsity is 0.1, and 200 subset samples in scenario 6, where sparsity is 0.05, to recover the true sparsity. Meanwhile, fullset gLasso needs at most 1,500 samples.

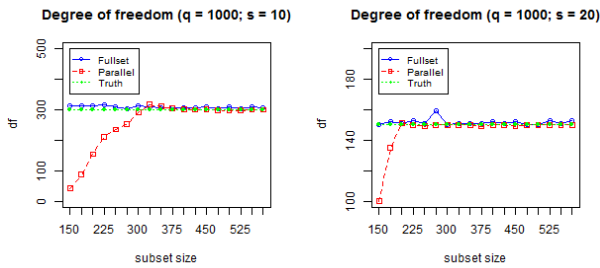


Figure 2: Sample complexity results in **Scenarios 5 & 6** (group-Lasso part is implemented by R-package *gglasso*).

### Performance with overlapping groups

To evaluate the performance of DC-ogLasso, we use a similar method of generating synthetic data as in [Lei *et al.*, 2013]. Consider the linear regression problem, where the nonzero components of the coefficient form a union of

groups. The group indices are predefined such that  $G_1 = \{1, 2, \dots, 10\}$ ,  $G_2 = \{6, 7, \dots, 15\}$ ,  $G_3 = \{11, 12, \dots, 20\}$ , ..., with each group having exactly 10 features and overlapping half of the previous group. Each feature group is selected in probability 0.1. The design matrix  $\mathbf{X}$  and selected components of the coefficient  $\beta$  are all generated from the multivariate  $\mathcal{N}(0, 1)$  without correlation. The standard gaussian noise  $\epsilon$  is generated with a scaling factor 0.01. Then the response vector  $Y$  is computed as in Eq. (1).

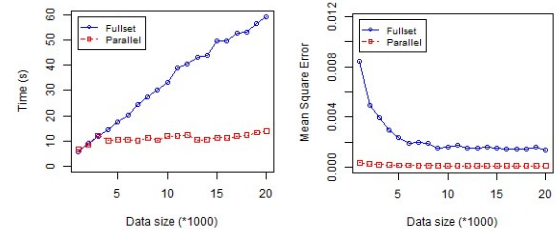


Figure 3: Performance of DC-ogLasso (overlapping group-Lasso part is implemented by R-package *grpregOverlap*)

The feature size is fixed ( $p = 1,000$ ) and sample size  $n$  varies from 1,000 to 20,000. For DC-ogLasso, each machine handles 1,000 samples. The result is shown in Fig. 3. Still, our DC method greatly shortens the computation time when  $n$  gets large. In terms of model selection consistency, when  $n \geq 2,000$ , DC-ogLasso using select-and-discard strategy has no less than 89 times out of 100 when it selects the correct model, while this number for overlapping group-lasso is 91. When using select-in-groups strategy, it drops to 87. In terms of the mean square error of the recovered model coefficient, DC-ogLasso outperforms overlapping group-lasso, thanks to the inclusion of the coefficient estimation stage.

### 5.2 MEMset Donor dataset

Groupwise inference methods like gLasso can be applied to predict donor splice sites, which play a key role in finding

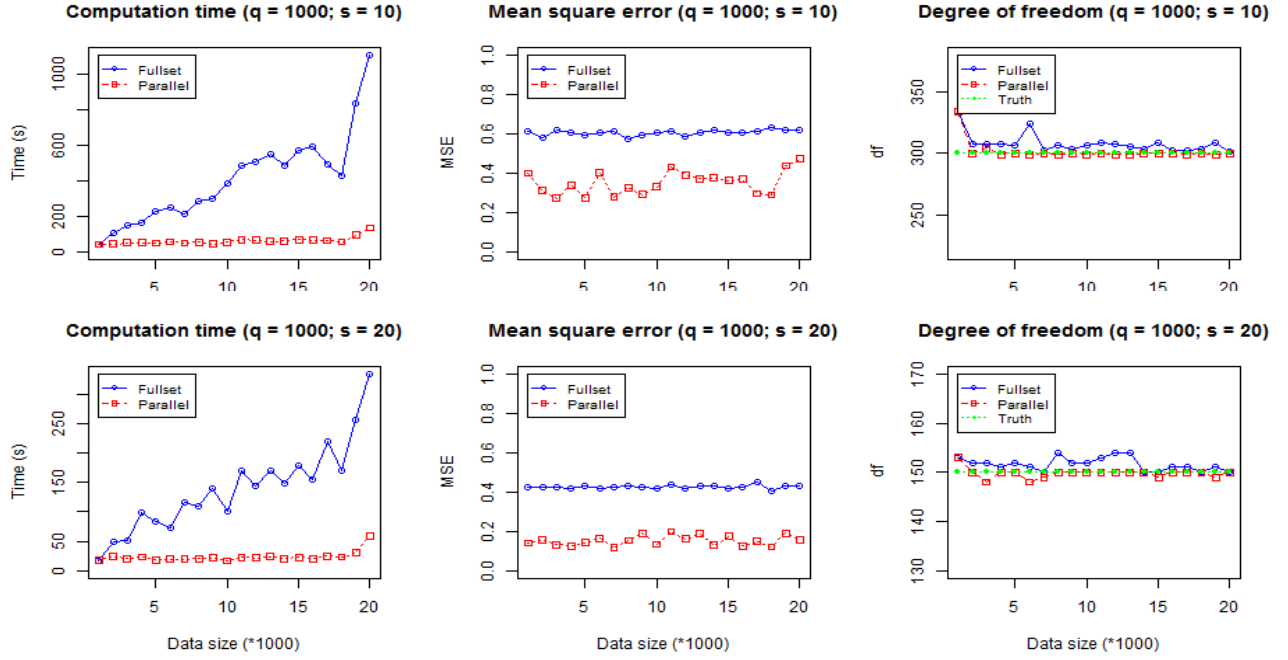


Figure 4: Results in **Scenarios 3 & 4** (group-Lasso part is implemented by R-package *gglasso*)

genes. The MEMset Donor dataset<sup>1</sup> is one of the splice sites datasets. It consists of a training set of 8,415 true and 179,438 false human donor sites with an additional test set of 4,208 true and 89,717 false donor sites. An instance of donor site sample is a sequence of 7 factors with four levels *A, C, G, T* (Please see Yeo and Burge [2004] for details). Here we follow an approach in [Meier *et al.*, 2008], which separates the original training set into a balanced training set of 5,610 true and 5,610 false donor sites, and a unbalanced validation set of 2,805 true and 59,804 false donor sites.

The data are represented as a collection of all factor interactions up to degree 2. Each interaction is encoded using dummy variables and treated as a group, leading to 63 groups of size varying from 4 to  $4^3$ , a total 2,604-dimension feature space. We train the model on the balanced training set, then choose the best threshold parameter  $\tau$  for classifying output over the trained model. That is, we assign sample  $i$  to the true class if  $p(x_i \in \text{true donor sites}) > \tau$  and to the false class otherwise. Finally, we evaluate the model using Pearson correlation between true class labels and predicted class labels on test set.

We compare the performance of logistic gLasso [Meier *et al.*, 2008] and logistic DC-gLasso on this dataset. Logistic DC-gLasso is a modified version of DC-gLasso on logistic regression. In logistic DC-gLasso, the square loss is replaced by the cross-entropy loss and the distributed least square regression in the coefficient estimation stage replaced by the distributed logistic regression.

Fig. 5 shows the performance of fullset logistic gLasso (denoted by "F") and logistic DC-gLasso using 2, 5 and 10 machines on the MEMset Donor dataset. In our experiment, the correlation results are 0.6598 for fullset inference, 0.6571,

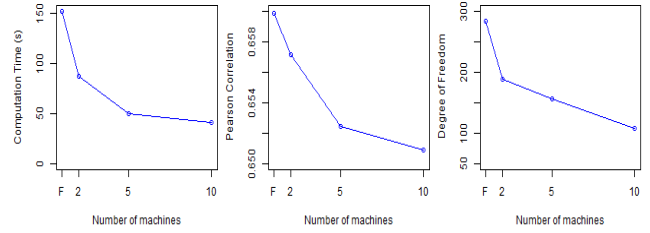


Figure 5: Results on the MEMset Donor dataset (group-Lasso part is implemented by R-package *gglasso*)

0.6524 and 0.6508 for logistic DC-gLasso on 2, 5 and 10 machines. Also note that the best reported result using logistic gLasso over degree 2 interactions is 0.6593 [Meier *et al.*, 2008]. Although our DC method suffers a small correlation loss compared with the original fullset approach, it enjoys a large reduce in training time: when running logistic DC-gLasso on 2 machines, only 87 seconds are needed to complete the original 151-second-job; when we increase the machine quantity to 5, it reduces to 50 seconds. In addition, the "Degree of Freedom" graph shows that the model we obtain shrinks as we split the training data in more parts, which makes sense because it's more difficult for more than half of all subsets to "vote" for one particular feature as the sample size in one subset declines, thus leading to a sparser model.

## 6 Conclusions

We propose DC-gLasso, a parallel inference method for group-Lasso and demonstrate its excellent performance on large data sets. DC-gLasso enjoys the advantage of parallel computing while minimizing the communication cost be-

<sup>1</sup>Available at <http://genes.mit.edu/burgelab/maxent/ssdata/>

tween machines. It successfully provides a scalable solution to the group-Lasso method: with enough machines, DC-gLasso can complete the task in a roughly constant short time regardless of the data size, with low mean square error and high probability of selecting the correct model.

## References

- [Argyriou *et al.*, 2011] Andreas Argyriou, Charles A Micchelli, Massimiliano Pontil, Lixin Shen, and Yuesheng Xu. Efficient first order methods for linear composite regularizers. *arXiv preprint arXiv:1104.1436*, 2011.
- [Bach, 2008] Francis R Bach. Consistency of the group lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.
- [Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.
- [Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
- [Hastie *et al.*, 2015] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.
- [Jacob *et al.*, 2009] Laurent Jacob, Guillaume Obozinski, and Jean Philippe Vert. Group lasso with overlap and graph lasso. In *International Conference on Machine Learning*, pages 433–440, 2009.
- [Lei *et al.*, 2013] Yuan Lei, Liu Jun, and Ye Jieping. Efficient methods for overlapping group lasso. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 35(9):2104 – 2116, 2013.
- [Lim and Hastie, 2013] Michael Lim and Trevor Hastie. Learning interactions through hierarchical group-lasso regularization. *arXiv preprint arXiv:1308.2719*, 2013.
- [Lim, 2013] Michael Lim. *The Group-lasso: Two Novel Applications*. PhD thesis, Stanford University, 2013.
- [Liu *et al.*, 2009] Jun Liu, Shuiwang Ji, Jieping Ye, et al. Slep: Sparse learning with efficient projections. *Arizona State University*, 6:491, 2009.
- [Mcdonald *et al.*, 2009] Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.
- [Meier *et al.*, 2008] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.
- [Ogutut and Piepho, 2014] Joseph O Ogutu and Hans-Peter Piepho. Regularized group regression methods for genomic prediction: Bridge, mcp, scad, group bridge, group lasso, sparse group lasso, group mcp and group scad. In *BMC proceedings*, volume 8, page S7. BioMed Central Ltd, 2014.
- [Peng *et al.*, 2013] Zhimin Peng, Ming Yan, and Wotao Yin. Parallel and distributed sparse optimization. In *Signals, Systems and Computers, 2013 Asilomar Conference on*, pages 659–646. IEEE, 2013.
- [Qin and Goldfarb, 2012] Zhiwei Qin and Donald Goldfarb. Structured sparsity via alternating direction methods. *The Journal of Machine Learning Research*, 13(1):1435–1468, 2012.
- [Qin *et al.*, 2013] Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.
- [Shimizu *et al.*, 2015] Yu Shimizu, Junichiro Yoshimoto, Shigeru Toki, Masahiro Takamura, Shinpei Yoshimura, Yasumasa Okamoto, Shigeto Yamawaki, and Kenji Doya. Toward probabilistic diagnosis and understanding of depression based on functional mri data analysis with logistic group lasso. 2015.
- [Simon *et al.*, 2013] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational & Graphical Statistics*, 22(2):231–245, 2013.
- [Souly and Shah, 2015] Nasim Souly and Mubarak Shah. Visual saliency detection using group lasso regularization in videos of natural scenes. *International Journal of Computer Vision*, pages 1–18, 2015.
- [Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [Villa *et al.*, 2014] Silvia Villa, Lorenzo Rosasco, Sofia Mosci, and Alessandro Verri. Proximal methods for the latent group lasso penalty. *Computational Optimization and Applications*, 58(2):381–407, 2014.
- [Wang *et al.*, 2014] Xiangyu Wang, Peichao Peng, and David B Dunson. Median selection subset aggregation for parallel inference. In *Advances in Neural Information Processing Systems*, pages 2195–2203, 2014.
- [Yang and Zou, 2014] Yi Yang and Hui Zou. A fast unified algorithm for solving group-lasso penalized learning problems. *Statistics and Computing*, pages 1–13, 2014.
- [Yeo and Burge, 2004] Gene Yeo and Christopher B Burge. Maximum entropy modeling of short sequence motifs with applications to rna splicing signals. *Journal of Computational Biology*, 11(2-3):377–394, 2004.
- [Yuan and Lin, 2006] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.
- [Zhang *et al.*, 2012] Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

- [Zhao and Yu, 2006] Peng Zhao and Bin Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7(3):2541–2563, 2006.
- [Zinkevich *et al.*, 2010] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.